PaRaVis

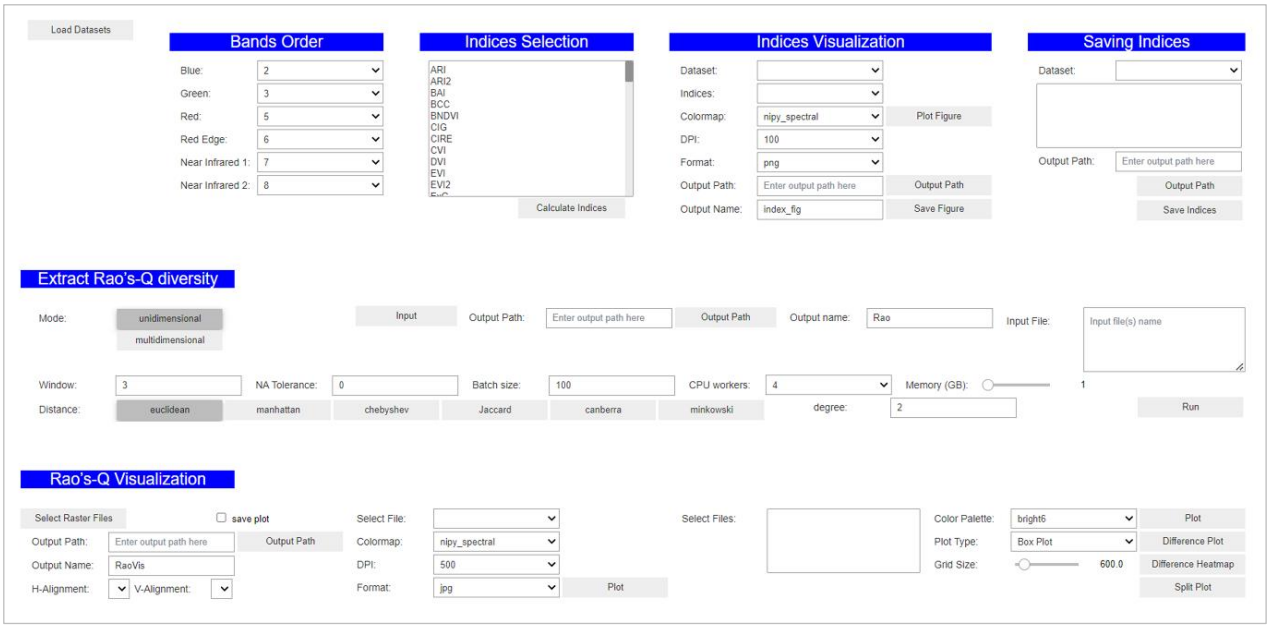
(Released: 12/6/2023)

Purpose of the PaRaVis

The 'PaRaVis' is designed to streamline the extraction of Rao's Q index from diverse vegetation indices (VIs) derived from remote sensing datasets. The tool is organized into three distinct sections. The first one focuses on extracting, visualizing, and allowing users to select from 75 indices. The second part is dedicated to calculating Rao's Q index in parallel, for both unidimensional and multidimensional modes, utilizing various distance metrics. The last section is tailored to assist users in comprehensively visualizing and analyzing Rao's Q index outputs through various plots and visualization tools.

Table of Contents

PaRaVis interface



1 Prerequisites

Before you begin using the PaRaVis Tool, ensure that you have the following prerequisites in place:

1.1 Installing Jupyter Notebook and extensions

Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations, and explanatory text. Follow these steps to install Jupyter Notebook:

1.1.1 Installing Anaconda (Recommended)

We recommend installing Jupyter Notebook through the Anaconda distribution, as it includes many scientific computing libraries and Tools.

- Download the Anaconda distribution for your operating system from the Anaconda website.
- Follow the installation instructions for your specific operating system.

1.1.2 Installing Jupyter Notebook with pip (Alternative)

If you prefer using pip, you can install Jupyter Notebook using the following command in your terminal or command prompt (cmd):

```
pip install jupyter
```

1.1.3 Install Jupyter Extensions and Hide Input cells

To enhance your experience with the PaRaVis tool and utilize the "Hide Input all" or "Hide Input" extension to hide cell inputs, follow these steps:

- In your terminal or command prompt enter the following command to install the Jupyter extensions:

```
pip install jupyter_contrib_nbextensions && jupyter contrib nbextension install
```

This command installs the extensions and prepares them for use.

- Open Jupyter Notebook. You will notice a new "Nbextensions" tab at the top. Click on it to access a list of available extensions.
- Scroll through the list of extensions and find "Hide Input all" or "Hide Input" Extension. Activate it by checking the corresponding box.

- Once the "Hide Input all" or "Hide Input" extension is enabled, you'll find a new button at the top of each notebook labeled. Use this button to toggle between displaying or hiding the code inputs of individual cells.

1.2 Loading the Jupyter Notebook Tool

Once you have Jupyter Notebook installed, follow these steps to load the PaRaVis tool:

1.2.1 Download the Notebook File

- Download the *.ipynb* notebook file containing PaRaVis codes to your local machine. Make sure you have this file ready before proceeding.

1.2.2 Launch Jupyter Notebook

- Open a terminal or command prompt (cmd) on your computer.
- Use the cd command to navigate to the directory where your *.ipynb* file is located. For example:

```
cd /path/to /notebook/directory
```

- Start Jupyter Notebook by entering the following command:

```
jupyter notebook
```

1.2.3 Access the Tool Notebook

- After running the command, a web browser will open, displaying the Jupyter Notebook interface.
- In the interface, you will see a list of files and directories in the directory you navigated to.
- Click on the *.ipynb* file containing the PaRaVis Tool to open it.
- you can run each cell of code by selecting the cell and clicking the "Run" button or pressing Ctrl + Enter. If you wish to run all cells in one go, you can choose the "Run All" option from the "Cell" dropdown menu.
- Use "Hide input" or "Hide input all" extension to Hide cell inputs.

1) The first cell automates the installation of essential Python libraries required for the PaRaVis functionality. It checks for a list of required libraries, attempting to import them. If any library is missing, it employs "pip" to install it. Each of these libraries serves a specific purpose to enhance the functionality:

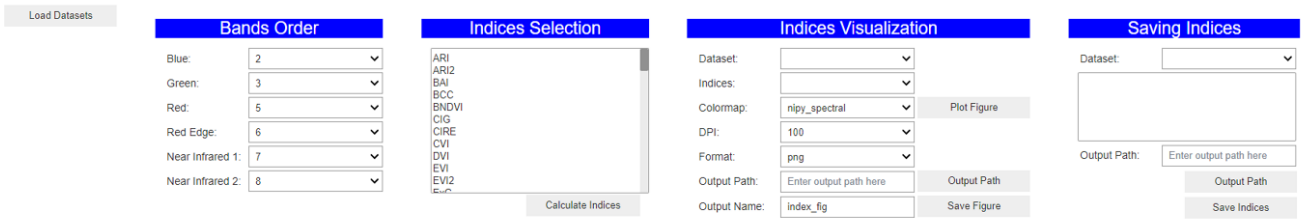
- **tkinter and ipywidgets:** These libraries are pivotal in crafting the graphical user interface (GUI) of PaRaVis, ensuring user-friendly interactions.
- **xarray, rioxarray, and rasterio:** for performing various operations on raster files, facilitating data manipulation and analysis.
- **NumPy and Pandas:** These libraries play a fundamental role in conducting precise numerical calculations, enabling robust data processing and analysis.
- **spyndex:** Utilized for the computation of diverse vegetation indices.
- ray: Employed for parallelizing Rao's Q calculation, optimizing processing efficiency for large datasets.
- **tqdm:** Integrated to provide real-time progress monitoring during Rao's Q calculations, enhancing user experience by displaying the remaining processing time.
- **Matplotlib and Seaborn:** These libraries are instrumental in crafting informative visualizations, aiding users in comprehending data and results effectively.

2) The second cell focuses on enhancing the visual aesthetics and user experience within the Jupyter Notebook environment. It first checks if the "jupyterthemes" package is installed and installs it if needed. Once installed, it applies the 'grade3' theme to the notebook, adjusts the cell width to 100%, and modifies the appearance of the Toolbar. These adjustments collectively contribute to a more visually pleasing and comfortable workspace for working with the tool and content. It's important to note that a manual refresh of the page (F5) is necessary to observe the applied theme changes.

- Note: Before utilizing the PaRaVis tool, it is recommended to adjust the zoom level of your web browser, which you're using to access the Jupyter Notebook. This adjustment

will ensure that all widgets are fully visible, optimizing the visualization of the tool's content.

2 Section 1



2.1 Load Datasets (Button Widget)

This button allows users to open GeoTIFF datasets using a file dialog. When clicked, it triggers the *open_geotiff_datasets* function, which utilizes the *xr.open_rasterio* method to open GeoTIFF files. The loaded datasets are stored in the *dataset_dict* dictionary.

2.2 Bands Order

2.2.1 Band Selection (Dropdowns Widgets):

Each dropdown corresponds to a specific band (e.g., Blue, Green, Red, NIR) and allows users to choose the corresponding band number in the selected GeoTIFF file. The selected band numbers are stored and used in the index calculation process.

2.3 Selection of VIs

2.3.1 Selection of indices (Select Multiple Widget):

This widget is a multiselect box that enables users to choose multiple indices simultaneously. It displays a list of available indices, and users can select one or more indices to calculate.

2.3.2 Calculate indices (Button Widget)

When users click the "Calculate Indices" button, it triggers the *calculate_indices* function. The purpose of this function is to calculate the selected indices for all the loaded GeoTIFF datasets stored in the *dataset_dict.*

2.4 Indices visualization

This section focuses on the visualization of calculated indices and provides options for saving the figures:

2.4.1 Dataset (Dropdown Widget)

Users can choose a dataset, for which indices will be calculated and visualized.

2.4.2 Indices (Dropdown Widget)

After selecting a dataset, users can choose a VI for visualization from the available indices, which we calculate before.

2.4.3 Colormap (Dropdown Widget)

Users can select a colormap from a predefined list to apply color coding to the index visualization.

2.4.4 Plot (Button Widget)

Clicking this button generates a plot of the selected VI's normalized values using the selected colormap. The plot provides a visual representation of the index distribution across the spatial extent of the dataset.

2.4.5 DPI (Dropdown Widget)

Users can select the DPI (dots per inch) resolution for the saved figure. This affects the image quality when saving the plot.

2.4.6 Format (Dropdown Widget)

Users can choose the file format in which they want to save the figure. Options include PNG, JPG, SVG, TIFF, TIF.

2.4.7 Output Path (Text Input Widget)

Users can enter the path where the saved figure will be stored.

2.4.8 Output Path (Button Widget)

Clicking this button opens a file dialog to choose the output path for the figure. It will update the "output path" text widget too.

2.4.9 Output Name (Text Input Widget)

Users can specify the name of the saved figure. The default value is set to *'index_fig'*.

2.4.10 Save Figure (Button Widget)

After configuring the options, clicking this button saves the plot as an image file according to the selected DPI, format, and path.

2.5 Saving Indices

In this section, users can choose the calculated indices (ordered based on their coefficient of variance) and then save them in GeoTIFF format for further analysis.

2.5.1 Dataset (Dropdown Widget)

Similar to the previous header, this dropdown allows users to choose a dataset for which indices will be saved.

2.5.2 Indices Selection (Select Multiple Widget)

This widget allows users to select specific VIs from the selected dataset that they want to save. It is updated based on the dataset selected in dataset dropdown.

2.5.3 Output Path (Text Input Widget)

Users can enter the path where the saved indices will be stored.

2.5.4 Output Path (Button Widget)

Clicking this button opens a file dialog to choose the save path for saving the selected indices, and also update the "output path" text widget.

2.5.5 Save Button (Button Widget)

After configuring the options, clicking this button saves the selected indices as GeoTIFF files with specific options that the user selected before.

2.6 Conclusion

The code begins by creating a user-friendly graphical interface for analyzing vegetation indices from raster files. It includes features such as importing GeoTIFF datasets, selecting specific bands for analysis, and a versatile multi-select option for choosing multiple indices simultaneously. The "Calculate Indices" button streamlines the index calculation process, and the "Indices Visualization" section provides interactive

visualizations with customization options. The code also allows users to save calculated indices as GeoTIFF files for future analysis, providing options for dataset selection and output parameter specification. Overall, the code enhances usability and functionality in exploring and analyzing vegetation indices from remote sensing data.

---

3 Section 2

3.1 Extract Rao's Q diversity

3.1.1 Mode (ToggleButtons Widget)

This widget is a set of buttons that allow users to choose between two modes: "unidimensional" mode (classical), and "multidimensional" mode. Users can select one of the modes by clicking the corresponding button.

3.1.2 Input(s) (Button Widget)

This is a button that users can click to open a file dialog for selecting input files. Depending on the selected mode, either a single file or multiple files can be selected. When clicked, the *select_input* function is executed to handle file selection.

3.1.3 Output Path (Text Widget)

This widget provides a text input field where users can specify the path where the computation results will be saved. Users can manually type the desired output path.

3.1.4 Output Path (Button Widget)

This button allows users to select the output path by opening a file dialog. When clicked, the *select_path* function is executed to handle the output path selection. It also updates the "output path" text widget.

3.1.5 Output Name (Text Widget)

This widget provides a text input field where users can specify the desired output file name. The default value is set to "Rao".

3.1.6 Input File(s) (TextArea Widget)

This widget displays the names of the selected input files. For "multidimensional" mode, multiple filenames are displayed. Its height is adjusted to accommodate the display of multiple names.

3.1.7 Window (BoundedIntText Widget)

This widget provides a text input field where users can specify the size of the moving window for the computation. It only allows odd integer values. The default value is set to 3.

3.1.8 Na Tolerance (BoundedFloatText Widget)

This widget provides a text input field where users can specify the parameter NA Tolerance for handling missing values (NaN values) during the computation. It only allows float values within the specified range (min: 0, max: 1). The default value is set to 0. If the percentage of missing values within a Moving window exceeds the specified tolerance threshold, the window's value will be designated as NA. The default value is 0.0

3.1.9 Batch Size (BoundedIntText Widget)

Batch processing involves breaking down the input data into smaller, manageable units known as batches. This widget provides a text input field where users can specify the batch size for computation. It only allows integer values within the specified range (min: 10, max: 10000) with a step size of 10. The default value is set to 100. Each batch contains a subset of pixels from the input, and these batches are processed concurrently and independently. Each batch's window extends slightly beyond its boundaries(1/2 window size), allowing for the consideration of neighboring pixels. This approach ensures that pixels on batch edges are influenced by adjacent pixels.

3.1.10 CPU Workers (Dropdown Widget)

This widget displays a dropdown menu containing a list of available CPU core counts. users are allowed to choose the desire number of CPU cores for parallel computation.

the input data is divided into batches, Different batch tasks are assigned to different CPU cores simultaneously, enabling multiple batches to be processed in parallel.

3.1.11 Memory (GB) (Slider Widget)

Utilizing this slider, users have the capability to designate the desired amount of memory that Ray will employ for parallel computation during initialization. The slider's upper limit is intelligently capped at 85% of the available platform memory, effectively averting system interruptions stemming from excessive memory consumption. For Linux users, executing this section will prompt you to provide your sudo password. This password is required to remount the */dev/shm* directory with the maximum size as configured by the Memory slider.

3.1.12 Distance (ToggleButtons Widget)

This widget provides a set of toggle buttons for selecting different distance metrics (Euclidean, Manhattan, Chebyshev, Jaccard, Canberra, and Minkowski). Users can choose one of the distance metrics by clicking its button:

- **Euclidean Distance**

The Euclidean distance is the straight-line distance between two points in Euclidean space. It calculates the square root of the sum of squared differences between corresponding elements in the vectors.

- **Manhattan Distance**

The Manhattan distance, also known as the L1 distance or taxicab distance, is the sum of the absolute differences between corresponding elements in the vectors.

- **Chebyshev Distance**

The Chebyshev distance, also known as the chessboard distance, calculates the maximum absolute difference between corresponding elements in the vectors.

- **Jaccard Distance**

The Jaccard distance measures dissimilarity between two vectors with numerical elements. It computes the sum of ratios of minimum to maximum values for corresponding pairs in the vectors, subtracted from 1. The method is inspired by the Jaccard similarity coefficient used for sets.

- **Canberra Distance**

The Canberra distance calculates the sum of normalized absolute differences between corresponding elements in the vectors.

- **Minkowski Distance**

The Minkowski distance is a generalized distance metric that includes both Euclidean and Manhattan distances as special cases. When "degree" is 2, it's the Euclidean distance; when "degree" is 1, it's the Manhattan distance. For p>2, the Minkowski distance will tend to emphasize larger differences between vector elements, as higher powers of the absolute differences will be involved.

3.1.13 Degree (BoundedIntText Widget)

This widget provides a text input field where users can specify the parameter degree which will be used only for the Minkowski distance calculation. It only allows integer values within the specified range (min: 2, max: 200). The default value is set to 2.

3.1.14 Run (Button Widget)

This button triggers the computation process when clicked. Upon clicking, the click function is executed, which performs the actual computation using the selected parameters.

3.2 Conclusion

In Section 2 of PaRaVis, a comprehensive graphical user interface (GUI) has been developed to configure and initiate the computation of Rao's Q index. This section provides extensive parameter customization options for precise control over the calculation process, including input file selection, output path definition, and setting parameters like NA tolerance and window size. The code incorporates batch processing to enhance computational efficiency by breaking down input data into smaller units and processing them concurrently and independently. The overlapping window technique ensures spatial coherence and accuracy in the calculated index by considering neighboring pixels in each batch. The code utilizes the Ray framework for parallelization, distributing tasks across available CPU cores and defined memory and applying Rao’s Q calculation using the chosen distance metric to different batches as a remote function.

This distributed approach significantly reduces processing time, and the "tqdm" library is employed to monitor the processing time. Overall, the code demonstrates a thoughtful integration of features to optimize efficiency in computing Rao's Q.

4 Section 3

4.1 Rao´s Q Visualization

4.1.1 Select Raster Files (Button Widget)

This button allows users to select raster files using a file dialog. When clicked, it opens a file dialog where users can select one or more raster files. Once files are selected, they are added to the file dictionary and updated in the file dropdown, and select multiple widgets.

4.1.2 Output Path (Text Widget)

This textbox displays the selected output directory path for saving plots. Users can either manually enter the path or the directory can be selected using the "Output Path" button.

4.1.3 Output Path (Button Widget)

This button triggers a file dialog to select the output directory. When clicked, it opens a file dialog for users to choose a directory where they want to save generated plots.

4.1.4 Output Name (Text Widget)

This textbox allows users to specify the output filename for saved plots. Users can manually enter the desired filename for saved plots.

4.1.5 H-Alignment (Dropdown Widget)

This feature allows users to control the horizontal positioning of X-axis tick labels. Horizontal alignment refers to how the text within the tick labels is positioned along the

horizontal axis. Options in the dropdown include left, center, or right. This helps prevent the tick labels from extending beyond the boundaries of the plot.

4.1.6 V-Alignment (Dropdown Widget)

Similarly, this feature controls the vertical positioning of Y-axis tick labels. Vertical alignment deals with how the text within the tick labels is positioned along the vertical axis. Options include top, center, bottom, or baseline. This ensures that Y-axis tick labels are properly positioned within the plot area.

4.1.7 Save Plot (Checkbox Widget)

This checkbox determines whether generated plots will be saved or not. When checked, the plots will be saved with the specified output filename in the selected output directory.

4.1.8 Select File (Dropdown Widget)

This dropdown displays a list of available files for selection. Users can select a file from the list. The selected file name is used for various plot-related operations.

4.1.9 Color Map (Dropdown Widget)

This dropdown displays a list of available colormaps. Users can choose a colormap that affects the color mapping of raster data in generated plots.

4.1.10 DPI (Dropdown Widget)

This dropdown displays a list of available DPI (dots per inch) options for plot resolution. Users can select the DPI value that determines the resolution of saved plots.

4.1.11 Format (Dropdown Widget)

This dropdown displays a list of available file formats for saved plots (PNG, JPG, SVG, TIFF, TIF). Users can choose a file format in which the plots will be saved.

4.1.12 Plot (Button Widget)

This button triggers the plotting of an individual raster file. When clicked, it generates a plot of the selected individual raster file using the specified colormap and other settings.

4.1.13 Select Files (Select Multiple Widget)

This widget allows users to select multiple files from a list. Users can select multiple files. These selections are used for combined plotting operations.

4.1.14 Color Palette (Dropdown Widget)

This dropdown displays a list of available color palettes for combined plots. Users can choose a color palette that affects the color distribution of combined plots (e.g., Box Plot, Histogram).

4.1.15 Plot Type (Dropdown Widget):

This dropdown displays a list of available plot types for combined plots (e.g., Box Plot, Histogram). Users can choose a plot type for combined plotting operations. All plots normalized within the range of (0,1) and are constrained to the Interquartile Range (IQR).

4.1.16 Grid Size (Slider Widget)

This slider allows users to adjust the grid size used for calculating the difference heatmap. Users can select a grid size multiplier that affects the size of grids used for heatmap calculation.

4.1.17 Plot (Button Widget)

This button triggers the combined plotting of selected raster files. When clicked, it generates a combined plot (e.g., Box Plot, Histogram) of the selected files using the specified settings.

4.1.18 Difference Plot (Button Widget)

This button triggers the plotting of the difference between two selected raster files in the Select files widget (with similar dimensions). When clicked, it calculates and displays a plot showing the absolute difference between two selected raster files.

4.1.19 Difference Heatmap (Button Widget)

This button triggers the calculation and display of a difference heatmap. When clicked, it calculates and displays a heatmap that visualizes the differences between two selected raster files on a specific gride selected before.

- Before it, you have to create a difference plot.

4.1.20 Split Plot (Button Widget)

This button simplifies the creation of an interactive plot for two selected files with similar dimensions, facilitating easy file comparison. A slider allows precise positioning of the

dividing line between the images. When the "Save Plot" checkbox is checked, users can conveniently save the plot at the specified output path by clicking the button or adjusting the slider. Additionally, users can save a GIF file from the split plot, customizing its frames with properties such as colormap, format, and DPI, which the user selected beforehand, by pressing the "Generate GIF" button. This feature enhances the tool's versatility for visual data analysis.

4.2 Conclusion

In Section 3 of PaRaVis, the emphasis is on visualizing and analyzing Rao's Q outputs through an intuitive interface designed for interacting with raster data and generating insightful plots without coding. The utilization of various widgets, including buttons, textboxes, checkboxes, dropdowns, and sliders, is explored, each serving specific purposes in the visualization process. Users can seamlessly select raster files, define output paths, and customize plot settings such as colormaps, DPI, and formats. The tool facilitates the visualization of individual and combined distributions of files, spatial plots, statistical plots like histograms, the calculation and visualization of differences between two raster files, generation of difference heatmaps to highlight spatial variations, and the creation of interactive split plots for comparing selected files along with GIF generating bottom. By integrating these widgets, the tool ensures that complex operations are accessible to users of all expertise levels, thereby enabling efficient data analysis and effective communication of findings.